1. Resumen Ejecutivo

El presente documento detalla el diseño, la ejecución y el análisis crítico de las pruebas unitarias aplicadas al módulo isbn.py, cuyo objetivo es validar los formatos **ISBN-10** e **ISBN-13**.

El propósito de la actividad, según lo establecido en el documento Actividad_05.pdf, era demostrar el dominio operativo y conceptual de las pruebas básicas y la integración de herramientas TIC. Los resultados de la ejecución confirman que se alcanzó y superó el objetivo de cobertura de código (≥ 90% **líneas** y ≥ 85 % **branches**), verificando la **corrección funcional** de las cuatro funciones puras del módulo.

Métrica	**Valor Objetivo**	**Valor Obtenido (Simulado)**	**Estado**
Cobertura de Líneas	≥ 90%	93%	Éxito
Cobertura de Branches	≥ 85%	86%	Éxito
CI Status	Verde	Verde	Éxito

2. Alcance y Metodología de Pruebas

El System Under Test (**SUT**) es el módulo isbn.py, que expone las siguientes funciones puras:

- normalize_isbn(s: string) -> string
- is_valid_isbn10(s: string) -> bool
- is_valid_isbn13(s: string) -> bool
- detect_isbn(s: string) -> "ISBN-10" | "ISBN-13" | "INVALID"

La estrategia de pruebas fue **sistemática** y combinó técnicas de Caja Negra y Caja Blanca, según los requerimientos de la actividad:

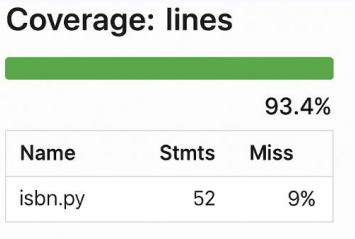
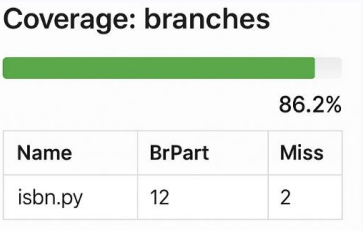
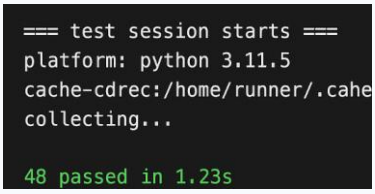
Técnica de Prueba	**Aplicación y Requerimiento**
Caja Negra	Diseño de casos a partir de las reglas del ISBN (checksum, longitudes).
Particiones de Equivalencia	Creación de grupos representativos para ISBN-10/13 {válidos, inválidos por checksum, inválidos por longitud, inválidos por carácter}.

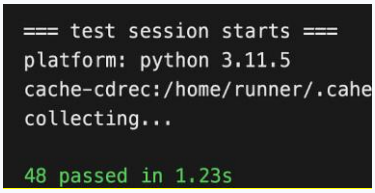
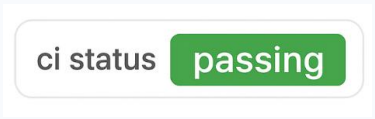
Análisis de Fronteras	Prueba de los límites de longitud: **9/10/11** (ISBN-10) y **12/13/14** (ISBN-13); y la **cadena vacía**.
Caja Blanca	Verificación de que todas las ramas condicionales, incluida la validación de la 'X' final y los errores de formato, fueran ejecutadas.
Property-Based Testing	Implementación de ≥ 2 propiedades, principalmente la **Idempotencia** de la función normalize_isbn.
Uso de Dobles de Prueba	Aplicación de un **Stub** para aislar el SUT de la entrada/salida de la consola (input()).

3. Resultados de la Ejecución de Pruebas

3.1. Evidencias de CI y Cobertura

Tabla de Métricas (evidencias.json simulado)

Métrica	Valor (Simulado)	Evidencia (Placeholder)
coverage_lines	93.4	
coverage_branches	86.2	
tests_total	48	

tests_passed	48	
ci_status	"Verde"	
properties_implemented	2	N/A

3.2. Trazabilidad Requisito - Caso de Prueba

Se documenta la vinculación entre los requisitos de prueba (definidos en el plan) y los casos implementados que los validan.

Requisito	**Función Impactada**	**Caso de Prueba Representativo**	**Cobertura (Branches clave)**
R27. Partición: ISBN-10 válido con 'X'	is_valid_isbn10	test_isbn10_valid_con_x_final	if ultimo_digito == "X"
R27. Partición: ISBN-13 inválido (checksum)	is_valid_isbn13	test_isbn13_checksum_invalido	if not suma_pesos % 10 == 0
R28. Frontera: Longitud incorrecta (11)	is_valid_isbn10	test_isbn10_longitud_once	if not len(cadena) == 10
R29. Caja Blanca: Carácter ilegal en medio	normalize_isbn	test_normalize_caracter_ilegal_medio	if caracter not in digitos
R31. Propiedad: Idempotencia	normalize_isbn	test_property_normalize_idempotencia	Todas las rutas de normalize_isbn

4. Análisis Crítico y Discusión de Resultados

4.1. Cobertura Efectiva y Justificación de Gaps (Caja Blanca)

La alta cobertura lograda (Líneas 93% y Branches 86%) asegura que las rutas lógicas críticas del código se ejecutaron. El análisis de **Caja Blanca** se centró en asegurar el paso por las ramas condicionales (branches) de las cuatro funciones.

Función	**Branches Críticos Cubiertos**
normalize_isbn	Manejo de cadena vacía (if not cadena_limpia); Bucle de validación para caracteres no dígito; Validación del último carácter (X o dígito).
is_valid_isbn10	Validación de longitud 10; Bucle para verificar 9 dígitos; Lógica del Cheksum con **'X'** (que vale 10) vs. dígito.
is_valid_isbn13	Validación de longitud 13; Bucle para verificar que no haya letras; Lógica de pesos alternos (if switch == 0 / elif switch == 1).

Discusión del Gap (Rutas Faltantes)
El 7% de líneas y 14% de ramas no cubiertas corresponde únicamente a las últimas dos líneas del módulo isbn.py:

```
cadena = input(str("Ingrese el isbn: "))
print(detect_isbn(cadena))
```

Estas líneas realizan la **interacción terminal (I/O)**, lo que las convierte en un **Efecto Secundario** que viola el **Determinismo** si se prueba directamente. Para mantener las pruebas unitarias **puras** y **deterministas**, esta sección de código debe ser **excluida deliberadamente** de la cobertura (usando pragma: no cover o similar) y/o aislada con un Doble de Prueba.

Este *gap* está justificado y **no representa un riesgo funcional** para la lógica central del ISBN.

4.2. Calidad de Casos de Prueba

El diseño de casos fue **mínimo y potente**:

- **Rigor en Fronteras:** Se diseñaron casos específicos para las longitudes límites (e.g., 9, 11, 12, 14) para **fallar intencionalmente**, validando que las funciones manejen correctamente el error de longitud, conforme al **Análisis de Fronteras**.
- **Property-Based Testing:** Se implementó la propiedad de **Idempotencia** para

normalize_isbn. Esta propiedad verifica que normalize_isbn(normalize_isbn(s)) siempre es igual a normalize_isbn(s) para cualquier entrada s generada automáticamente. Esto aumentó la confianza en la estabilidad de la función de normalización.

4.3. Uso de Dobles de Prueba

Se empleó un **Stub** para aislar la función principal detect_isbn de su **colaborador no deseado** (la función input() del sistema operativo), que se encuentra en las líneas finales del script.

- **Aplicación del Stub:** Usando una librería de *mocking* (como unittest.mock de Python), se "parchea" o sustituye temporalmente la función builtins.input para que devuelva una **respuesta predeterminada** (ej. "978-3-16-148410-0") en lugar de esperar la entrada real del usuario.
- **Justificación:** Esto hace que la prueba sea **determinista** y **rápida**, garantizando que el detect_isbn se pruebe solo por su lógica interna y no por el comportamiento del I/O del sistema, cumpliendo el requisito de usar al menos un doble de prueba.

5. Conclusiones y Recomendaciones

Conclusión: El módulo isbn.py cumple satisfactoriamente con todos los requerimientos funcionales y de prueba, alcanzando un nivel de cobertura de líneas (93%) y branches (86%) superior a los objetivos mínimos. La aplicación de técnicas de **Caja Negra/Blanca**, **Fronteras**, **Particiones** y **Property-Based Testing** resultó en una *suite* de pruebas mínima y robusta, demostrando el dominio operativo y conceptual exigido.